ELSEVIER

Discrete Applied Mathematics 91 (1999) 25–38

DISCRETE APPLIED MATHEMATICS

Minimal connected enclosures on an embedded planar graph

Christos Alexopoulos [a,1], J. Scott Provan [b,*], H. Donald Ratliff [a,2], Bryan R. Stutzman [c,2]

[a] *School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205, USA*

[b] *Department of Operations Research, University of North Carolina, Chapel Hill, NC 27599-3180, USA*

[c] *CAPS Logistics, Inc., 2700 Cumberland Parkway, Atlanta, GA 30339, USA*

Received 5 April 1996; received in revised form 4 May 1998; accepted 18 May 1998

Abstract

We study five problems of finding minimal enclosures comprised of elements of a connected, planar graph with a plane embedding. The first three problems consider the identification of a shortest enclosing walk, cycle or trail surrounding a polygonal, simply connected obstacle on the plane. We propose polynomial algorithms that improve on existing algorithms. The last two problems consider the formation of minimal zones (sets of adjacent regions such that any pair of points in a zone can be connected by a non-zero width curve that lies entirely in the zone). Specifically, we assume that the regions of the graph have nonnegative weights and seek the formation of minimum weight zones containing a set of points or a set of regions. We prove that the last two problems are NP-hard and transform them to Steiner arborescence/fixed-charge flow problems. © 1999 Published by Elsevier Science B.V. All rights reserved.

1. Introduction

The purpose of this paper is to develop algorithms for combining regions formed by embedded planar graphs. Planar graphs are used to represent many systems, with transportation networks (e.g., roads, rivers, rail) being examples. There are a variety of sources, including the US government, for such databases. In these networks, edges represent transportation links augmented with additional edges for natural boundaries (e.g., rivers), man-made boundaries (e.g., power lines), and political boundaries (e.g., county lines), and vertices are formed from the intersections of these elements. Our

* Corresponding author. E-mail: scott_provan@unc.edu. Supported in part by NSF grant No. CCR-9200572.

[1] Supported in part by AFOSR grant No. F49620-93-1-0043.

[2] Supported in part by ONR grant No. N00014-95-1-0380.

0166-218X/99/$ – see front matter © 1999 Published by Elsevier Science B.V. All rights reserved.
PII: S0166-218X(98)00095-X

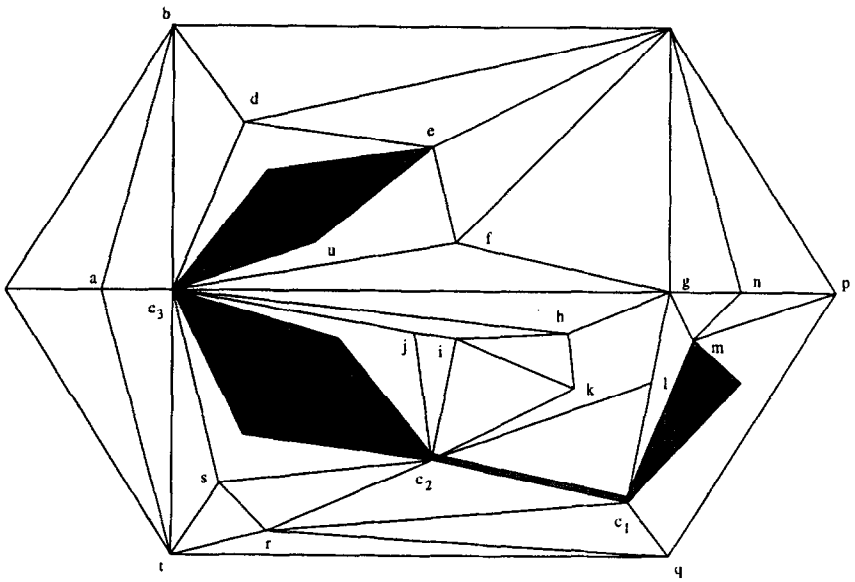

Fig. 1. A plane graph with shaded obstacle.

work is motivated by computer applications in the areas of Distribution, Logistics and Geographic Information Systems.

We start with some basic definitions. We consider a connected planar graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ with a plane embedding (or simply a *plane* graph) and assume that every edge $e\in\mathcal{E}$ is a straight line segment with a positive length (weight) $l(e)$. Let O be an obstacle on the plane (see Fig. 1). For technical convenience, we assume that O is simply connected with polygonal boundary; O may, however, have "zero-width" sections, where a local portion of the obstacle consists of a single edge (as in the edge (c_1,c_2) of the obstacle in Fig. 1). We can also assume that O has no edges of $\mathcal{G}$ in its interior, since these edges are irrelevant for the purposes considered in this paper, and can be removed. The vertices and edges defining the boundary of O may be coincident with vertices and edges of $\mathcal{G}$, although we will consider the edges of $\mathcal{G}$ independently of the obstacle boundary. (In Fig. 1 we take all edges of O to be in G.)

An *O-enclosing* walk is any closed path in $\mathcal{G}$, with possibly repeated nodes or edges, that "goes completely around" O, or more technically, cannot be homotopically shrunk to a point without crossing O. An *O-enclosing trail* is an O-enclosing walk that does not repeat an edge, and a *O-enclosing cycle* is a O-enclosing walk that does not repeat an edge or a vertex. In Fig. 1, for example, $c_3deuc_3jc_2c_1lgnpqc_1c_2sc_3$ is an O-enclosing walk, $c_3deuc_3jc_2khgnpqc_1c_2sc_3$ is an O-enclosing trail, and $c_3defgnpqc_1c_2sc_3$ is an O-enclosing cycle. If the edges of the closed path coincide with the obstacle we consider the obstacle to lie to the "inside" of the path edge, and in the case of a repeated edge coincident with a "zero-width" section of O, the obstacle is considered to lie "between" the two traversals of the edge.

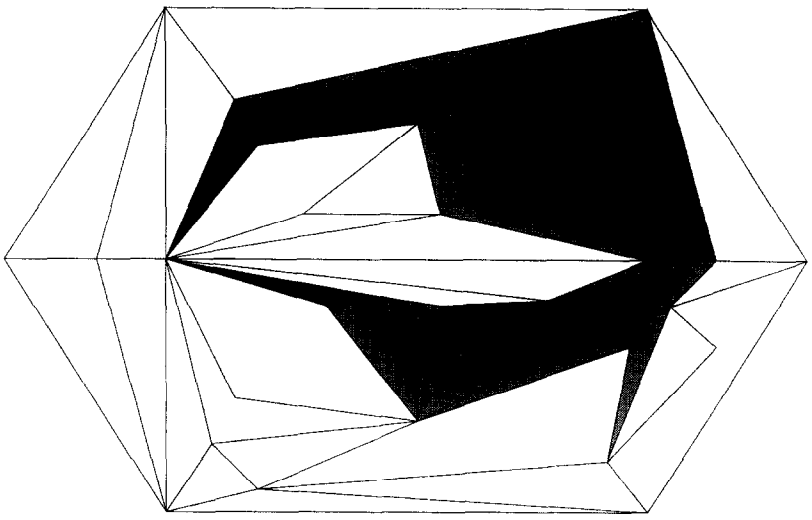

Fig. 2. Shaded zone surrounding four regions.

A *region* of $\mathcal{G}$ is the closure of a face, that is, the union of the face and its boundary. We denote the *unbounded* region that lies outside the graph by r_0 and the bounded regions by $r_1, r_2, \ldots, r_k$. Let $\mathcal{R} = \{r_0, r_1, \ldots, r_k\}$. A region r_j may have a non-negative *weight* (such as population or perimeter) denoted by $w(r_j)$. A *zone* is a set of adjacent regions such that any pair of points in the set can be connected by a non-zero width curve that lies entirely in the zone (i.e., every point on the curve has an ε-neighborhood that is a subset of the zone). In Fig. 2, the shaded area represents a zone containing the four darkened regions.

In this paper, we consider the following five problems:

Shortest Enclosing Walk/Cycle/Trail (SEW/SEC/SET) Problem

Instance: A plane graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edges $e \in \mathcal{E}$ having weights $l(e) > 0$, and polygonal obstacle O.

Find: Walk/cycle/trail Γ enclosing O of minimum total length $\sum_{e \in \Gamma} l(e)$.

Minimum Weight Zone Containing a Set of Regions (MWZR) Problem

Instance: A plane graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with regions r_i having weights $w(r_i) > 0$ and a set of *terminal* regions $\mathcal{R}^* \subseteq \mathcal{R}$ in $\mathcal{G}$.

Find: A zone $\mathcal{Z}$ that contains all terminal regions and has minimum total weight $\sum_{r_i \in \mathcal{Z}} w(r_i)$.

Minimum Weight Zone Containing a Set of Points (MWZP) Problem

Instance: A plane graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with regions r_i having weights $w(r_i) > 0$ and a set $\mathcal{V}^*$ of *terminal* points on the plane.

Find: A zone $\mathcal{Z}$ that contains all terminal points and has minimum total weight $\sum_{r_i \in \mathcal{Z}} w(r_i)$.

The SEW, SEC, and SET problems arise when buildings on a street network are to be surrounded by a security fence laid on the streets with minimum length [3]. The SEC and SET problems have an additional interesting application in constructing minimum weight 2-connected networks spanning a given set of points [12]. The MWZP and MWZR problems have applications in political districting where an objective is the formation of political districts by combining precincts based on their weights (such as population or percentage of minority voters).

Remark 1. A problem similar to MWZP is the Fixed Embedding Face Cover Problem [2]: Given a plane graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ together with a subset of terminal vertices $\mathcal{V}^*\subset\mathcal{V}$, find the minimum weight covering of $\mathcal{V}^*$ by regions. Unlike MWZP, however, the Fixed Embedding Face Cover Problem does not require that the regions be connected.

Bienstock and Monma [3] gave the first algorithm to solve the SEW problem on a plane graph, by solving a flow problem on a nonplanar augmentation of the dual graph. Provan [11] proposed polynomial algorithms for both the SEW and SEC problems on general graphs with a given (not necessarily plane) layout. All three algorithms are $O(|\mathcal{V}|^2\log|\mathcal{V}|)$; however, the algorithm for SEC given in [11] does not always give an optimal solution, as discovered by Stutzman in [14]. Bienstock and Monma [2] proposed algorithms for the Fixed Embedding Face Cover Problem; these, however, do not apply directly to the zone problems.

In this paper we develop algorithms for, and investigate the complexity of, the SEW, SEC, SET, MWZP, and MWZR problems. Section 2 describes an $O(|\mathcal{V}|\log|\mathcal{V}|)$ algorithm for solving SEW by perturbing the obstacle and solving a max flow in the planar dual graph. Section 3 gives a (correct) algorithm for the SEC problem having complexity $O(|\mathcal{V}|)$, and shows how it can be modified to solve the SET problem. Section 4 studies the problems MWZP and MWZR. Both problems are shown to NP-hard, but it is shown how they can be reduced to a special case of the directed Steiner tree problem.

2. Shortest enclosing walk

In this section, we review the literature on the computation of a SEW around an obstacle O on the plane and propose some improvements on the algorithms of [3, 11]. A *cutvertex* of O is a vertex of O whose removal will disconnect O. Let $c_1,\ldots,c_k$ be the cutvertices of O. Fig. 1 depicts a plane graph with shaded obstacle O having cutvertices c_1, c_2 and c_3.

We begin by describing Algorithm Perturb which produces the *perturbed* graph $\hat{\mathcal{G}}$ by duplicating the cutvertices of O and moving them "outward" from O. The perturbed graph will remain plane, and the perturbed obstacle $\hat{O}$ will have no cutvertices, or equivalently, its boundary will be a simple polygon. Moreover, the

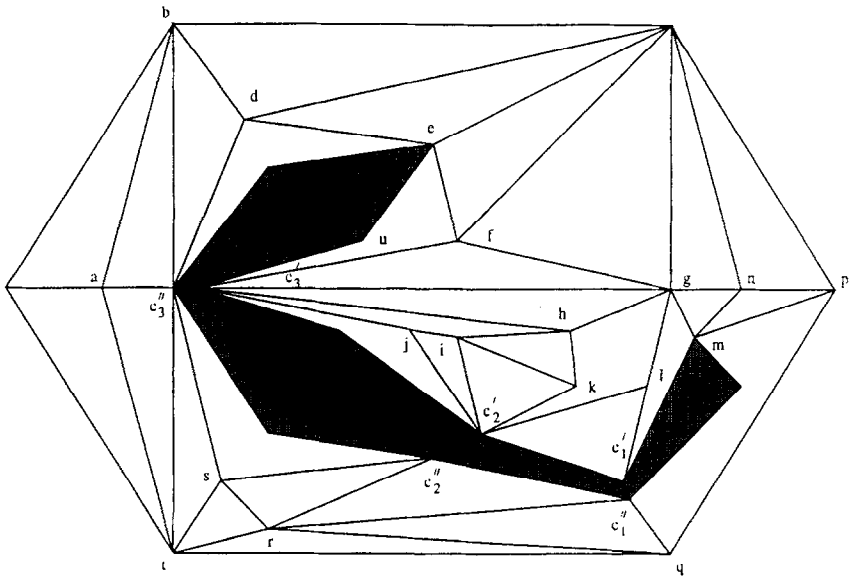

Fig. 3. The perturbed graph $\hat{\mathcal{G}}$.

corresponding O-enclosing walks in $\mathcal{G}$ and $\hat{O}$-enclosing walks in $\hat{\mathcal{G}}$ will have the same lengths.

Algorithm Perturb

1. Let $c_1,\ldots,c_k$ be the cutvertices of O. Set $\hat{\mathcal{V}}=\mathcal{V}$, $\hat{\mathcal{E}}=\mathcal{E}$, and $\hat{O}=O$.
2. **For** $i=1,\ldots,k$ **do**
3. Let $v_1,\ldots,v_{m_i}$ be the vertices of O adjacent to c_i in the clockwise direction. Let $\angle v_j c_i v_{j+1}$, $j=1,\ldots,m_i$ be the angles in the plane with vertex c_i and no edges of O incident to c_i in their interior.
4. **For** $j=1,\ldots,m_i$ **do**
5. Place a vertex c_{ij} in the interior of $\angle v_j c_i v_{j+1}$ sufficiently near c_i.
6. Replace every edge (c_i,u) in the closure of $\angle v_j c_i v_{j+1}$ by a new edge (c_{ij},u) with length equal to that of (c_i,u).
7. Remove vertex c_i from $\hat{\mathcal{V}}$.
End for
End for
8. Delete the elements of $\hat{\mathcal{G}}$ inside $\hat{O}$.
End

Fig. 3 shows the perturbed graph $\hat{\mathcal{G}}$ for the instance given in Fig. 1. It is clear that the resulting graph remains plane. Now let $\hat{\mathcal{G}}_D$ be the *dual* graph of $\hat{\mathcal{G}}$, that is, $\hat{\mathcal{G}}_D$ has a vertex inside each region of $\hat{\mathcal{G}}$ — including vertex v_0 corresponding to the unbounded region r_0 and vertex v_* corresponding to the region r_* containing

$\hat{O}$ — and every edge e of $\hat{\mathcal{G}}$ corresponds to edge e_D of $\hat{\mathcal{G}}_D$ connecting the vertices corresponding to the two regions of $\hat{\mathcal{G}}$ adjacent to e. Give each edge of $\hat{\mathcal{G}}_D$ the weight of the corresponding edge of $\hat{\mathcal{G}}$. It is a well-known fact that there is a one-to-one correspondence between (v_0, v_*)-cuts in $\hat{\mathcal{G}}_D$ and cycles in $\hat{\mathcal{G}}$ separating r_0 from r_*. Since $\hat{O}$-enclosing walks in $\hat{\mathcal{G}}$ must be cycles separating r_0 from r_*, the length of such a cycle will be equal to the weight of the corresponding (v_0, v_*)-cut in $\hat{\mathcal{G}}_D$. Hence, the minimum weight (v_0, v_*)-cut in $\hat{\mathcal{G}}_D$ corresponds to a shortest $\hat{O}$-enclosing walk in $\hat{\mathcal{G}}$. Since the perturbed graph $\hat{\mathcal{G}}$ and its dual can be formed in $O(|\mathcal{V}|)$ time, and finding a minimum (r_0, r^*)-cut in $\hat{\mathcal{G}}_D$ takes $O(|\mathcal{R}| \log |\mathcal{R}|) = O(|\mathcal{V}| \log |\mathcal{V}|)$ time using the planar max flow algorithm of Frederickson [7], then the SEW problem can be solved in $O(|\mathcal{V}| \log |\mathcal{V}|)$ time, an improvement on the $O(|\mathcal{V}^2| \log |\mathcal{V}|)$ time algorithms given in [3, 11].

Remark 2. An important extension of the SEW problem is that of enclosing a *collection* $\mathcal{O}^*$ of disconnected obstacles by a shortest closed walk. Bienstock and Monma [3] did this in the case where the obstacles are single points; the extension to the general case is fairly straightforward (and simplifies the procedure given in [11]). First contract each of the individual obstacles in $\mathcal{O}^*$ to a single point, and then find a tree of shortest paths from an arbitrarily chosen one of these contracted obstacles to each of the other contracted obstacles. If we let O' be the union of the elements of $\mathcal{O}^*$ together with the edges of the shortest path tree, then any SEW for O' will be an SEW surrounding $\mathcal{O}^*$. The shortest path tree can be found in $O(|\mathcal{V}| \log |\mathcal{V}|)$ time by means of Johnson's algorithm with a Fibonacci heap [4, p. 565], and by using the perturbation method given above to find the SEW for O', we obtain an SEW for $\mathcal{O}^*$ in $O(|\mathcal{V}| \log |\mathcal{V}|)$ time.

Provan [11] gives an algorithm for the SEW problem for general (nonplane) embeddings of G which has as a subroutine an $O(|\mathcal{V}| \log |\mathcal{V}|)$ algorithm for finding an SEW for obstacle $\hat{O}$ going through a particular point s. First perturb the obstacle as in the previous section, and fix arbitrary point p in the interior of the perturbed obstacle $\hat{O}$. For every edge $e = (u, v)$ of $\hat{\mathcal{G}}$, we can compute the two angles $\theta(u, v)$ and $\theta(v, u) = -\theta(u, v)$ of clockwise sweep from p when traversing e from u to v and from v to u, respectively. The *winding angle* of a walk Γ in $\hat{\mathcal{G}}$, say $\theta(\Gamma)$, is obtained by summing the angles $\theta(u, v)$ for each edge (u, v) of Γ traversed in the direction u to v. It is well known ([1, Section 4.2]) that a closed walk encloses $\hat{O}$ if and only if it has a nonzero winding angle, which in the plane graph case corresponds to winding angle 2π. The algorithm SEW(s) given below finds an SEW for $\hat{O}$ passing through the vertex s.

Algorithm SEW(s)

1. Find a shortest path tree in $\hat{\mathcal{G}}$ rooted at s. For each $v \in \hat{\mathcal{V}}$, let Γ_{sv} be the path from s to v on the tree, and let Γ_{sv}^{-1} be the path Γ_{sv} traversed in the reverse direction. Let $d(s, v)$ be the length of Γ_{sv}.
2. Find an SEW for $\hat{O}$ by minimizing $d(s, u) + l(u, v) + d(s, v)$ over the set of edges $(u, v) \in \hat{\mathcal{E}}$ such that $\theta(\Gamma_{su}) + \theta(u, v) + \theta(\Gamma_{sv}^{-1}) = 2\pi$.

Again, the shortest path tree in Step 1 can be found in $O(|\mathscr{V}'|\log|\mathscr{V}'|)$, and Step 2 can be performed in $O(|E|)=O(|V|)$ time, for a total complexity of $O(|\mathscr{V}'|\log|\mathscr{V}'|)$. The SEW problem can then be solved in time $O(|\mathscr{V}'|^2\log|\mathscr{V}'|)$ by repeated application of SEW(s). SEW(s) will also be useful in the next section for solving the SEC problem.

3. Shortest enclosing cycle

The SEC problem of finding a shortest enclosing cycle differs from SEW in that we are not allowed to repeat any vertices. Provan [11] describes an algorithm that takes as input a shortest enclosing walk and then progressively adds regions until a cycle is found. Although his algorithm always finds an enclosing cycle if one exists, this cycle is not always the shortest such enclosing cycle (see [14, pp. 49–51]). In this section, we propose an algorithm which correctly finds the SEC.

First form the perturbed graph $\hat{\mathscr{G}}$ with respect to O by applying Algorithm Perturb as given in Section 2. The boundary of the perturbed region $\hat{O}$ is now a simple polygon, denoted by $\hat{W}_O$, and the cutvertices of O now have multiple corresponding copies in $\hat{\mathscr{G}}$. Thus in order for an $\hat{O}$-enclosing walk in $\hat{\mathscr{G}}$ to correspond to a cycle in $\mathscr{G}$, it will be necessary to prevent it from having more than one copy of any cutvertex of O. For any two vertices u_1, u_2 on $\hat{W}_O$, we denote by $[u_1, u_2]$ the path going clockwise on $\hat{W}_O$ from u_1 to u_2.

Next order the cutvertices of O as $c_1,\ldots,c_k$ so that for $i<j<l$ the vertex c_i does not separate c_j from c_l in O. Note that the vertices c_1, c_2 and c_3 in Fig. 1 are ordered according to this criterion; in fact any order not starting with c_2 will satisfy the criterion.

Lemma 1. *Let the cutvertices of O be ordered $c_1,\ldots,c_k$ as above. Let $i<j<l$, and let c_j' and c_l' be copies in $\hat{\mathscr{G}}$ of vertices c_j and c_l, respectively. Then the path $[c_j', c_l']$ contains either* all *copies of c_i or* no *copies of c_i.*

Proof. From the requirement on processing order we know that c_i cannot be on a path from c_j to c_l in O, that is, c_i is separated from both c_j and c_l by a cutvertex v of O (possibly $v=c_j$ or $v=c_l$). From the construction of $\hat{O}$, however, it is clear that if $[c_j', c_l']$ passes through a copy of v into the portion of O containing c_i, then it cannot again encounter v without having circumscribed this entire portion of O, thereby passing through all copies of c_i. Since neither c_j' nor c_l' is in this portion of O, then $[c_j', c_l']$ either contains all copies of c_i, or it contains none of them. □

Remark 3. Lemma 1 provides a linear-time method for putting the cutvertices in the order required. Simply traverse $\hat{W}_O$ clockwise, adding each vertex to the list at the point of the traversal at which all of its copies have been visited. In the example in Fig. 3, if we began the clockwise traversal at vertex m, we would visit the copies of the cutvertices in the order c_1'', c_2'', c_3'', c_3', c_2', c_1'. The resulting cutvertex ordering would then be c_3, c_2, c_1.

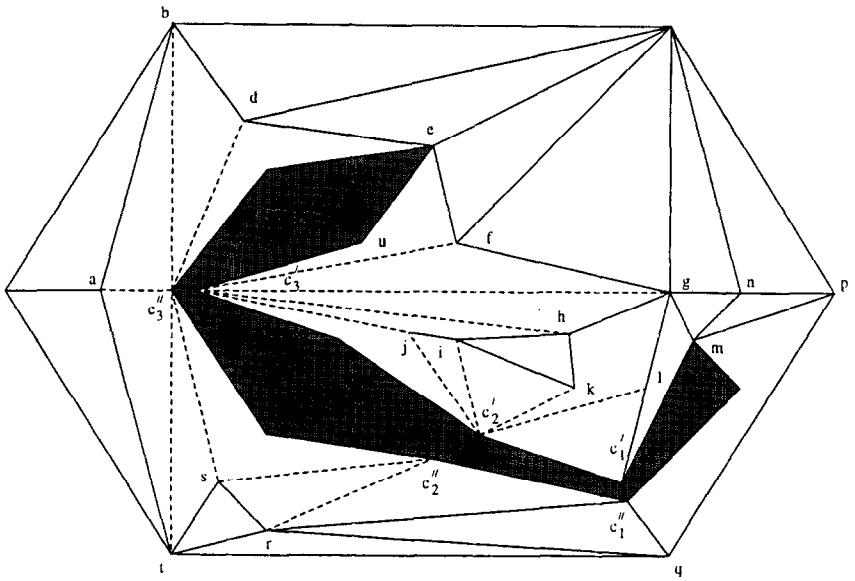

Fig. 4. The graph $\hat{\mathscr{G}}_1$ after processing of vertex c_1 (edges of $\hat{\mathscr{G}}\backslash\hat{\mathscr{G}}_1$ are dashed).

The SEC algorithm first processes the cutvertices in the order $c_1,\ldots,c_k$ given above. At Stage i of the algorithm, let $P_i=\{c_1,\ldots,c_i\}$ be the set of processed vertices and let $U_i=\{c_{i+1},\ldots,c_k\}$ be the set of unprocessed vertices, with $\hat{P}_i$ and $\hat{U}_i$ the corresponding sets of copies in $\hat{\mathscr{G}}$. Define the graph $\hat{\mathscr{G}}_i$ to be the graph obtained from $\hat{\mathscr{G}}$ by removing the vertices of $\hat{U}_i$ and their adjacent edges. Fig. 4 shows the graph $\hat{\mathscr{G}}_1$ obtained by removing the set $\hat{U}_1=\{c'_2,c''_2,c'_3,c''_3\}$ from $\mathscr{G}$ in Fig. 3. We will find the SEC solution by successively updating information about "partially enclosing paths" in $\hat{\mathscr{G}}_i$. Let e_1 and e_2 be two edges in $\hat{\mathscr{G}}\backslash\hat{\mathscr{G}}_i$, so that edge e_j is adjacent to a vertex $u_j\in\hat{U}_i$, $j=1,2$. A *P_i-clockwise (e_1,e_2)-walk* is a (u_1,u_2)-walk Γ in $\hat{\mathscr{G}}_i\cup\{e_1,e_2\}$ satisfying

(i) e_1 and e_2 are the initial and final edges of Γ;
(ii) $\Gamma\cup[u_2,u_1]$ encloses $\hat{O}$ (equivalently, Γ has the same winding angle as $[u_1,u_2]$);
(iii) $\Gamma\cap[u_1,u_2]$ contains at most one copy of any vertex in P_i;
(iv) $\Gamma\cap[u_2,u_1]$ contains *no* copies of any vertex in P_i.

For example, consider the graph $\hat{\mathscr{G}}_1$ given in Fig. 4, and let $u_1=c'_2$, $u_2=c'_3$, $e_1=(c'_2,k)$, and $e_2=(c'_3,h)$. Then an example of a P_2-clockwise (e_1,e_2)-walk in $\hat{\mathscr{G}}_1$ is the walk $c'_2khgnpqc''_1rtabdefghc'_3$. For computational purposes we will allow a P_i-clockwise (e_1,e_2)-walk to contain repeated edges, although it will become clear that no such walks can contribute to the final shortest enclosing cycle. We denote the length of a shortest P_i-clockwise (e_1,e_2)-walk by $\delta_i(e_1,e_2)$, with $\delta_i(e_1,e_2)=\infty$ if no such path exists.

We start by showing how to compute δ_0. Let $e_1=(u_1,v_1)$ and $e_2=(u_2,v_2)$ be edges of $\hat{\mathscr{G}}\backslash\hat{\mathscr{G}}_0$, with $u_1,u_2\in\hat{U}_0$. Add to $\hat{\mathscr{G}}_0$ the edges e_1 and e_2, together with the edges

in $[u_2, u_1]$. Set the weights of the edges in $[u_2, u_1]$ to 0, and use Algorithm SEW(u_1) from Section 2 to compute a shortest $\hat{O}$-enclosing walk Γ_0 in this graph going through the point u_1. Note that such a path must contain e_1, e_2, and all edges of $[u_2, u_1]$. Set $\Gamma = \Gamma_0 \backslash [u_2, u_1]$. Then Γ immediately satisfies (i) and (ii), and since $P_0 = \emptyset$ then (iii) and (iv) automatically hold.

We next show how δ_{i+1} can be recursively derived from δ_i.

Lemma 2. *Suppose $\delta_i(\,)$ is known, and let $e_1 = (u_1, v_1)$, $e_2 = (u_2, v_2) \in \hat{\mathscr{G}} \backslash \hat{\mathscr{G}}_{i+1}$ with $u_1, u_2 \in \hat{U}_{i+1}$. Then*

$$\delta_{i+1}(e_1, e_2) = \min(\{\delta_i(e_1, e_2)\} \cup \{\delta_i(e_1, f_1) + \delta_i(f_2, e_2)\colon f_1, f_2 \text{ adjacent to a copy } c'_{i+1} \text{ of } c_{i+1} \text{ lying in } [u_1, u_2]\}). \quad (*)$$

Proof. ($\geqslant$) Let Γ be a P_{i+1}-clockwise (e_1, e_2)-walk of length $\delta_{i+1}(e_1, e_2)$. If Γ does not contain the point c_{i+1} then Γ must also be a P_i-clockwise (e_1, e_2)-walk, and hence is of length at least $\delta_i(e_1, e_2)$. Otherwise, let c'_{i+1} be the single copy of c_{i+1} contained in Γ, and note that condition (iv) implies that $c'_{i+1} \in [u_1, u_2]$. Make the edges of Γ distinct by copying duplicated edges of Γ and placing them side by side in arbitrary order. Now consider the Eulerian plane graph H of edges consisting of Γ (with duplicated edges) together with the edges of $[u_2, u_1]$ (which by assumption are distinct from Γ). By property (ii) we know that $\hat{O}$ lies in an interior face of H, and the boundary of this face consists of a closed walk W having no duplicate edges and containing $[u_2, u_1]$ as a contiguous set of edges (since these always lie inside any edges of $\hat{\mathscr{G}}$). Consider the path $\Gamma' = W \backslash [u_1, u_2] \subseteq \Gamma$. By construction $\Gamma' \cup [u_2, u_1]$ encloses $\hat{O}$. Further, Γ' must contain all vertices of Γ on $\hat{O}$, in particular, Γ' contains u_1, u_2, and c'_{i+1}. It must therefore contain e_1 and e_2 as well, since these are the only edges adjacent to u_1 and u_2 on Γ. Now c'_{i+1} splits Γ' into two edge-disjoint noncrossing paths Γ_1 and Γ_2, with Γ_1 starting at e_1 and ending at some edge f_1 adjacent to c'_{i+1}, and Γ_2 starting at some edge f_2 adjacent to c'_{i+1} and ending at e_2. We show that Γ_1 is a P_i-clockwise (e_1, f_1)-walk, the proof that Γ_2 is a P_i-clockwise (f_2, e_2)-walk being similar. Since $[c'_{i+1}, u_2]$ lies on the "inside" of W, then Γ_2 can be replaced by $[c'_{i-1}, u_2]$ in W and still enclose $\hat{O}$. This implies that $\Gamma_1 \cup [c'_{i-1}, u_1]$ encloses $\hat{O}$, and so (ii) holds. Since (iii) holds for Γ and $\Gamma_1 \subset \Gamma$, then (iii) holds for Γ_1. For (iv), first observe that Γ_1 does not cross either $[u_1, c'_{i+1}]$ or Γ_2, and so the only vertices of $[c'_{i-1}, u_2]$ in Γ_1 are also in Γ_2. Further, these vertices cannot also be in P_i, since Γ satisfies (iii). Second, observe that Γ satisfies (iv), so that Γ_1 can never contain a vertex of P_i in $[u_2, u_1]$. These two observations together imply that Γ_1 satisfies (iv) as well.

We have established that Γ_1 is a P_i-clockwise (e_1, f_1)-walk, and (similarly) Γ_2 is a P_i-clockwise (f_2, e_2)-walk. Thus Γ_1 has length at least $\delta_i(e_1, f_1)$ and Γ_2 has length at least $\delta_i(f_2, e_2)$. It follows that Γ has length at least that of Γ', whose length in turn is at least $\delta_i(e_1, f_1) + \delta_i(f_2, e_2)$.

($\leqslant$) First note that any P_i-clockwise (e_1, e_2)-walk is also a P_{i+1}-clockwise (e_1, e_2)-walk, since the extra vertex c_{i+1} included in P_{i+1} has no copies in $\hat{\mathscr{G}}_i$. Therefore

$\delta_{i+1}(e_1,e_2)\leqslant\delta_i(e_1,e_2)$. Now choose a copy c'_{i+1} of c_{i+1} lying in $[u_1,u_2]$ together with adjacent edges f_1 and f_2. Let Γ_1 and Γ_2 be P_i-clockwise (e_1,f_1)- and (f_2,e_2)-walks of length $\delta_i(e_1,f_1)$ and $\delta_i(f_2,e_2)$, respectively. Then $\Gamma=\Gamma_1\cup\Gamma_2$ is a (u_1,u_2)-walk with initial and final edges e_1 and e_2, and so (i) holds. $\Gamma\cup[u_2,u_1]$ encloses $\hat{O}$, since its winding angle is the sum of the winding angles of Γ_1 and Γ_2, and so (ii) holds. Also, $\Gamma\cap[u_2,u_1]$ cannot contain any elements of P_{i+1}, since neither $\Gamma_1\cap[c'_{i+1},u_1]$ nor $\Gamma_2\cap[u_2,c'_{i+1}]$ does, and so (iv) holds. Finally, Lemma 1 implies that since u_1, u_2, and c'_{i+1} are all copies of unprocessed vertices, then for any vertex $c_j\in P_i$, if a copy of c_j lies in $[u_1,c'_{i+1}]$ then no copy of c_j lies in $[c'_{i+1},u_2]$. Since (iv) holds for both Γ_1 and Γ_2, it follows that they cannot both contain copies of the same element of P_i, and so (iii) holds for Γ.

We have established that Γ is a P_{i+1}-clockwise (e_1,e_2)-walk, and hence has length at least $\delta_{i+1}(e_1,e_2)$. Thus $\delta_{i+1}(e_1,e_2)\leqslant\delta_i(e_1,f_1)+\delta_i(f_2,e_2)$, and the lemma follows. □

We are now in position to describe the SEC algorithm.

Algorithm SEC

1. Find the 2-connected components and cutvertices of O. Order the cutvertices as $c_1,\ldots,c_k$ so that for $i<j<l$ the vertex c_i is not on a path in O connecting c_j and c_l.
2. Construct the perturbed graph $\hat{\mathscr{G}}=\hat{\mathscr{G}}_0$, and find the length $\hat{\delta}$ of an SEW for O. If the SEW is a cycle, set $\delta^*=\hat{\delta}$ and stop.
3. Compute $\delta_0(e_1,e_2)$ for all $e_1=(u_1,v_1)$, $e_2=(u_2,v_2)\in\hat{\mathscr{G}}\backslash\hat{\mathscr{G}}_0$ with $u_1,u_2\in\hat{U}_0$.
4. **For** $i=0,\ldots,k-1$ **do**
   For each pair $e_1,e_2\in\hat{\mathscr{G}}\backslash\hat{\mathscr{G}}_i$, compute the lengths $\delta_{i+1}(e_1,e_2)$, using $(*)$.
   Set $\tilde{\delta}_{i+1}=\min\{\delta_{i+1}(e_1,e_2)\colon e_1,e_2$ adjacent to the same copy c'_{i+1} of $c_{i+1}\}$.
   End for
5. The length of an SEC for O in $\mathscr{G}$ is $\delta^*=\min\{\tilde{\delta}_0,\ldots,\tilde{\delta}_k\}$.
   End

The actual SEC can be found by keeping track of the SEW for O in $\hat{\mathscr{G}}_0$ and the individual P_i-clockwise (e_1,e_2)-walks.

Theorem 1. *Algorithm* SEC *finds a shortest enclosing cycle for* $\hat{O}$ *in* $O(|\mathscr{V}|^4)$ *time.*

Proof. For the correctness of the algorithm, first note that the $\delta_i()$ values are correctly computed by Lemma 2 and the preceding discussion. Next, observe that all of the walks found in either Step 2 or 4 of the algorithm enclose O and contain no duplicate copies of a cutvertex of O. Further, any shortest (and hence edge-minimal) walk satisfying these two properties must be a cycle. On the other hand any enclosing cycle for O in $\mathscr{G}$ either *never* hits a cutvertex of O — in which case it corresponds to an SEW

for O in $\mathcal{G}_0$ as found in Step 2 — or else it hits O in some cutvertex c_i of highest index i — in which case it corresponds to a P_i-clockwise (e_1, e_2)-walk as found Steps 3 or 4. Thus, the shortest of the walks found in Steps 2–4 is an SEC for O in $\mathcal{G}$.

The complexity is dominated by Step 4, which requires $k = O(|\mathcal{V}|)$ stages. Let m_i be the number of vertices adjacent to cutvertex c_i, $i = 1, \ldots, k$. Then stage i of Step 4 requires $O((\sum_{j=i}^{k} m_j)^2)$ evaluations of $(*)$, which in turn has $O(m_i^2)$ terms. Hence, stage i takes $O(m_i^2(\sum_{j=i}^{k} m_j)^2)$ time. As a result, the total time complexity of the algorithm is $O(\sum_{i=1}^{k} m_i^2(\sum_{j=i}^{k} m_j)^2) = O(|\mathcal{V}|^4)$ since $\sum_{i=1}^{k} m_i^2(\sum_{j=i}^{k} m_j)^2 \leqslant (\sum_{i=1}^{k} m_i^2)(\sum_{j=1}^{k} m_j)^2$ and $\sum_{i=1}^{k} m_i^2 \leqslant (\sum_{j=1}^{k} m_j)^2 = O(|\mathcal{V}|^2)$. □

Remark 4. We can solve SEC for a *set* $\mathcal{O}^*$ of polygonal obstacles just as was done for SEW: simply solve the SEC problem on obstacle O' as constructed in Remark 2. Since an SEC must enclose the innermost SEW, then the optimal cycle enclosing $\mathcal{O}^*$ must also enclose O', and so is the SEC for $\mathcal{O}^*$.

Remark 5. We can also easily modify Algorithm SEC to solve the Shortest Enclosing Trail problem. First bisect each bridge e of the connected obstacle by adding a new vertex v_e. Now apply Algorithm Perturb to the obstacle, but leave the bisecting vertices v_e unperturbed. The SEC for the resulting obstacle can now repeat vertices of the original obstacle, but will repeat no edges, because it cannot repeat any of the bisecting vertices. The modification when enclosing a set of obstacles proceeds just as in Remarks 2 and 4.

Remark 6. The SEC and SET problems have the following interesting application. Let K be a set of "terminals" lying on (the boundary of) a region of $\mathcal{G}$. Then the minimum weight 2-edge-connected (2-vertex-connected) subgraph of $\mathcal{G}$ which spans K is an SET (SEC) for K. It follows that we can find a minimum weight 2-edge-connected (2-vertex-connected) subgraph of $\mathcal{G}$ spanning K in $O(|\mathcal{V}|^4)$ time. See [12] for details.

4. Minimum weight zone problems

In this section we study the zone formation problems MWZR and MWZP. These problems are closely related to versions of the *Steiner tree* problem in graphs (for a good account of this area, see [13]). We start by considering the following version of the Steiner tree problem.

Planar Node-Weighted Steiner Tree (PVST) Problem

Instance: Planar graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with *vertices* $v \in \mathcal{V}$ having weights $c(v) > 0$, and specified subset $\mathcal{K}$ of vertices of $\mathcal{G}$.

Find: Tree $\mathcal{T}$ of edges of $\mathcal{G}$ that spans all vertices in $\mathcal{K}$ and has minimum total weight $\sum_{v \in \mathcal{T}} c(v)$.

PVST was shown to be NP-hard in [8], even when all vertex weights are 1. (The problem studied in [8] actually involves finding the tree with minimum *edge* cardinality, but this number is always exactly 1 less than the vertex cardinality of the tree.) The relationship between PVST and MWZR is as follows.

Lemma 3. *Let $\mathcal{G}$ be a plane graph and $\mathcal{K}$ a set of vertices of $\mathcal{G}$. Let $\mathcal{G}_D$ be the dual graph to $\mathcal{G}$, and let $\mathcal{R}^*$ be the set of regions dual to the vertices of $\mathcal{K}$. Then the solutions to PVST in $\mathcal{G}$ are dual to the solutions to MWZR in $\mathcal{G}_D$.*

The proof is straightforward. □

Theorem 2. *The problems MWZR and MWZP are NP-hard.*

Proof. The reduction of PVST to MZWR follows from Lemma 3. To reduce MZWR to MZWP, simply place an isolated vertex of $\mathcal{V}^*$ in the center of each terminal region. Then the zones containing the vertices of $\mathcal{V}^*$ are precisely those containing the regions of $\mathcal{R}^*$. □

Lemma 3 also gives the *reverse* reduction from MWZR to PVST. Steiner tree problems have had an extensive amount of research (again, see [13]), much of which is applicable to the problems studied here. In particular, the minimum *cardinality* version of the problem (where all regions have weight 1) can be solved using any of the numerous standard Steiner tree solution techniques. The general node-weighted version of the problem, moreover, can be reduced to the following version of the Steiner tree problem.

Steiner Arborescence (SA) Problem

Instance: A *directed* graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ with edges $e\in\mathcal{E}$ having weights $l(e)>0$, *source* vertex s, and set $\mathcal{D}$ of *demand* vertices.

Find: Tree $\mathcal{T}$ that admits (directed) paths from s to every vertex in $\mathcal{D}$ and has minimum weight $\sum_{e\in\mathcal{T}} l(e)$.

Chapter II.6.2 of [13] gives the reduction of PVST to SA, by simply replacing each undirected edge with two directed edges having weights equal to the vertex weight of their heads, choosing any vertex $s\in\mathcal{K}$ as the source, and setting $\mathcal{D}$ to be the remaining vertices of $\mathcal{K}$. It follows that MWZR can be solved by solving an SA problem, the weights differing by exactly $c(s)$. The MWZP problem can also be reduced to SA by creating the weighted directed graph as above, choosing any vertex $s\in\mathcal{V}^*$ as the source, and then adding edges directed *from* s to the dual vertex of each region containing s, and *into* each vertex $v\in\mathcal{V}^*\backslash s$ from the dual vertex of each region containing that vertex. The weight of each edge out of s is assigned the weight of its head plus M, for a sufficiently large number M, and the remaining added edges have

weights 0. Setting $\mathscr{K} = \mathscr{V}^* \backslash s$, we get that any Steiner arborescence for this problem corresponds to a zone containing $\mathscr{V}^*$ of the same weight, plus M. (The large weights on edges out of s ensure that only one edge out of s will be used, thereby preventing s from being a cutvertex for the zone.)

Many of the techniques for the Steiner tree problem apply to the SA problem as well. This includes some powerful network and linear programming methods involving its relationship to the *single-source uncapacitated fixed-charge flow problem*. Nemhauser and Wolsey [10, pp. 495–512] review a variety of algorithms for solving fixed-charge network flow problems. A heuristic solution of MWZR/MWZP can be obtained in polynomial time by linearizing the fixed costs on the edges. As well, there are three network-specific techniques that provide polynomial-time algorithms for restricted instances of the Steiner tree problem, and which also apply to the SA problem. In particular, the MZWR problem can be solved in polynomial time for any of the following restricted classes of instances:

1. The number of terminal regions is bounded above by some fixed integer [5].
2. The number of regions that are *not* terminal is bounded above by some fixed integer [9].
3. All of the terminal regions contain a common vertex of $\mathscr{G}$ [6].
4. There is a fixed number of vertices of $\mathscr{G}$ such that each terminal region contains at least one of these vertices [6].

The MZWP problem can be solved in polynomial time for any of the following restricted classes of instances:

1. The number of terminal vertices is bounded above by some fixed integer [5].
2. All of the vertices are contained in a set of terminal regions containing a common vertex of $\mathscr{G}$ [6].

References

[1] L.A. Alfors, Complex Analysis, McGraw-Hill, New York, 1966.
[2] D. Bienstock, C.L. Monma, On the complexity of covering vertices by faces in a planar graph, SIAM J. Comput. 17(1) (1988) 53–76.
[3] D. Bienstock, C.L. Monma, Optimal enclosing regions in planar graphs, Networks 19(1) (1989) 79–94.
[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Press, Cambridge, MA, 1990.
[5] S.E. Dreyfus, R.A. Wagner, The Steiner problem in graphs, Networks 2 (1972) 195–207.
[6] R.E. Erickson, C.L. Monma, A.F. Veinott, The send-and-split method for minimum-concave-cost network flows, Math. Oper. Res. 12 (1987) 634–644.
[7] G.N. Frederickson, Fast algorithms for shortest paths in planar graphs, with applications, SIAM J. Comput. 16(6) (1987) 1004–1022.
[8] M.R. Garey, D.S. Johnson, The rectilinear Steiner tree problem is NP-complete, SIAM J. Appl. Math. 32 (1977) 826–834.
[9] S.L. Hakimi, Steiner's problem in graphs and its applications, Networks 1 (1971) 113–133.
[10] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Programming, Wiley, New York, 1988.

[11] J.S. Provan, Shortest enclosing walks and cycles in embedded graphs, Inform. Process. Lett. 30 (1989) 119–125.
[12] J.S. Provan, On finding two-connected subgraphs in planar graphs, Oper. Res. Lett. 20 (1997) 81–84.
[13] D.S. Richards, F.K. Hwang, P. Winter, The Steiner Tree Problem, North-Holland, New York, 1992.
[14] B.R. Stutzman, Zone formation problems on embedded planar graphs, Ph.D. Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 1992.